Challenges and Experiences with Applying Microsoft Threat Modeling in Agile Development Projects

Daniela S. Cruzes
SINTEF Digital
Trondheim, Norway
daniela.s.cruzes@sintef.no

Martin Gilje Jaatun
SINTEF Digital
Trondheim, Norway
martin.g.jaatun@sintef.no

Karin Bernsmed
SINTEF Digital
Trondheim, Norway
karin.bernsmed@sintef.no

Inger Anne Tøndel
NTNU
Trondheim, Norway
inger.anne.tondel@ntnu.no

***Abstract*— The goal of secure software engineering is to create software that keeps performing as intended even when exposed to attacks. Threat modeling is considered to be a key activity, but can be challenging to perform for developers, and even more so in agile software development. Hence, threat modeling has not seen widespread use in agile software projects. The goal of this paper is to investigate the challenges facing adoption of threat modeling using the Microsoft approach with STRIDE. We performed a case study in a company comprising five agile development projects. We identified 21 challenges to threat modeling that emerged from our observations. We then mapped these challenges to challenges found in the literature. Some challenges overlap the findings from the literature; the extra challenges we have found in our exploratory study came mostly from the activities of asset identification and also from our observations on what happened after the threat modeling meetings. This study shows that we still have to address many challenges in order to get a proper adoption of threat modeling in agile development projects.**

Keywords—STRIDE, Agile Software Development, Threat Modeling, Software Security, Secure Software Engineering

I. Introduction

The goal of secure software engineering is to create software that keeps performing as intended even when exposed to attacks. This is more than just resilience; "as intended" also includes maintaining required levels of confidentiality and integrity, not just availability. Threat modeling is considered to be a key activity in achieving this goal. The main purpose of threat modeling is to identify and mitigate potential risks by means of eliciting or refining security requirements. Threat modeling is a process by which potential threats (such as structural vulnerabilities) can be identified, enumerated, and prioritized, simulating an attacker's perspective. Such activities often take place in the design phase, and are repeated later on during the product life-cycle, if necessary.

The threat modeling activity is particularly important in software security, since many security vulnerabilities are caused due to architectural design flaws [1]. Furthermore, fixing such vulnerabilities after implementation may be very costly, requiring workarounds which sometimes increase the attack surface. A well-defined threat model helps to identify threats to different assets of a system by utilizing well-grounded assumptions on the capabilities of any attacker interested in exploiting such a system. It also enables the development teams to identify critical areas of the design which need to be protected, as well as mitigation strategies. However, threat modeling can also be challenging to perform for developers, and even more so in agile software development.

Various threat modeling approaches and methodologies have been developed over time and they have been used in the process of designing secure applications, varying from conceptual frameworks to practical methodologies [2]–[4]. In a recent systematic review, Tuma et al. [2] highlight that the existing threat modeling techniques have insufficient quality assurance of the outcomes, use merely validation by illustration to evaluate their proposed technique, and in addition most approaches do not provide tool support.

In agile software development, adoption of security practices poses different challenges, often because security activities are not prioritized, or because the practitioners are not able to see the relevance and importance of the activities to the improvement of the security in the project [5]. The same holds true for threat modeling; the practice is not widespread, and the agile practitioners have few sources of recommendations on how to proceed to adopt the practice in their development process [6]–[9]. Researchers have not yet gathered enough evidence on how to include threat modeling in software development processes that do not rely on having up-front design for their solutions. Studies in software security usually focus on software security activities in general, and there are few empirical studies focusing on specific practices in agile software development.

This research aims to build evidence on the adoption of threat modeling using the Microsoft Threat Modeling Technique with STRIDE in agile projects; focusing on the challenges and experiences of different projects in one company where we have been facilitating the adoption of threat modeling in agile software development. This study is part of a research project which investigates how to meaningfully integrate software security into agile software development activities. The project started in October 2015 and will end in October 2020. The company of our case study is a company with which we have had long-term collaboration in multiple projects since 2013.

The remainder of this paper is structured as follows: In Section II we discuss related work, and Section III covers the methodology and context of our study. In section IV we present the results with the mapping to the related work. In section V we discuss our results by stating the implications to research and practice. In Section VI we describe some

limitations of our study, and section VII contains the conclusions and future work.

II. Background

Threat modeling is used to identify threats which may be related to specific vulnerabilities. If it is not feasible to remove the vulnerabilities, additional security controls that prevent those vulnerabilities from being exploited must be added.

There are various research papers that explore threat modeling in different domains [10]–[17]. Tuma et al. [2] recently conducted a systematic literature review (SLR) of the existing techniques for threat modeling. In their study they analyzed 38 primary studies where a total of 26 techniques were compared with respect to their applicability, characteristics of the required input for analysis, characteristics of analysis procedure, characteristics of analysis outcomes, and ease of adoption. The most commonly used techniques in the presented body of knowledge were misuse cases, attack trees, problem frames and several software-centric approaches that are well recognized in the software engineering community, particularly in the industrial space, such as STRIDE [18], [19], CORAS [20], and P.A.S.T.A [21]. The most frequently used technique in industry is STRIDE [22].

Traditionally, threat modeling activities are coupled to the different phases of the waterfall workflow, starting with a global view of the ultimate upfront design. In contrast, agile is much more incremental, and service-oriented development lacks centralized control. Several proposals have appeared over the years on how to integrate security engineering practices into agile methodologies like Scrum and XP [23], built on the common premise that agile teams neglect security because it is not an explicit part of common agile frameworks. Poller et al. [24] suggest that Scrum works well as a management model, and security development requires iterations as in agile development, yet Scrum teams can fail to address security needs due to their low visibility, competing objectives, and Scrum's division of labor.

Türpe and Poller [23] theorize about tensions between the characteristics of security requirements and security work on the one hand, and the way Scrum manages development work on the other. Analyzing the definition of Scrum, the authors find three different ways of managing security work: as bug fixing on demand, continuously as a quality requirement through the definition of "done," or as prioritized and planned development work through the product backlog. The authors discuss the capabilities and limitations of these approaches and find each of them inadequate. On-demand fixing rarely leads to substantial security improvement. As a quality requirement, security has a complex relationship with development work and is difficult to verify. Security features in the backlog would be a suitable approach to many security concerns, but they compete with other requirements and may also need special expertise to design and implement effectively. Türpe and Poller also argue that research aiming to reconcile security engineering with agile development should consider not only the execution of security activities in an agile process, but also the challenges of managing security work in agile frameworks. Their analysis suggests four areas of security tasks that are worth investigating and supporting: the reflective discovery of security needs to create backlog items, the valuation and prioritization of security work, agile verification and feedback in the security dimension, and the collaboration of Scrum teams with external security experts and consultants.

Gálvez and Gürses [25] analyze which challenges and opportunities the shifts in software engineering practice introduce into traditional Threat Modeling activities; how they relate to the different Privacy Goals; and which Agile principles and Service properties have an impact on them. The authors mention that the following principles from the agile manifesto are relevant to the threat modeling process [25]:

1) The system design must be flexible enough to accommodate new requirements anytime. Software will be developed iteratively, implementing new requirements in each sprint while maintaining existing functionality.
2) Working software should be delivered frequently. Customer needs are assumed to evolve through the use of systems. Frequent delivery allows developers to address them in fixed-timed iterations where priorities are set based on feedback from the customer.
3) To transfer knowledge and convey critical information, developers should prioritize face-to-face meetings rather than read-only documentation. This preference for oral communications is expected to enhance the global understanding of the system and diminish the need for detailed diagrams and exhaustive reports.
4) Working software is the primary measure of progress, and Test Driven Development enables developers to show what is already done, and third parties to check that it is done correctly. Together with face-to-face meetings, working software has been shown to be a more effective vehicle of communication than written documentation.

Gálvez and Gürses [25] presented a list of 21 challenges and opportunities that the shifts in software engineering practice introduce into traditional Threat Modeling activities (as shown in Table 2); how they relate to the different Privacy Goals; and what Agile principles and Service properties have an impact on them.

Tuma et al. [2] provide insight into the obstacles for adopting the existing approaches, and discuss the current state of their adoption in software engineering trends (e.g. Agile, DevOps). For them, there are four important aspects where existing threat modeling techniques have yet to mature: (i) traceability of analysis in the code base; (ii) composability of analysis outcomes; (iii) threat impact analysis automation; and (iv) definition of done [2]:

- T1 - There is a need for highly automated threat modeling techniques due to short sprints. In the space of threat approaches, tools have been used for three main purposes: i) partially automating the analysis procedure, ii) graphically representing threats to the system and (iii) facilitating the analysis execution (i.e. helping the analyst to follow the procedure).
- T2 - It is important that the information that was gained from threat modeling is automatically propagated to source code level (and vice-versa);
- T3 - The existing techniques would benefit from guidelines of how to compose the analysis outcome, since the software systems under analysis in practice are too large and complex to be analyzed all at once; the analysis performed for one subsystem is related to security assumptions which may not be in line with the security assumptions of another subsystem.

- T4 - Analysts are also faced with the challenge of deciding how many identified threats (and at what level of abstraction) are enough for a "good" analysis of a particular subsystem.

Finally, as part of our previous work, Tøndel et al. [4] have performed a case study in a university setting comprising several agile development projects to learn more about challenges facing adoption of the threat modeling game Microsoft Elevation of Privilege (EoP). On the adoption of EoP in Agile, the authors concluded that EoP has the potential to improve security interest and awareness of the team participants, and can be useful for training in threat modeling; the system needs to be complex enough to have a ROI of the effort; there is a need to make the game more fun and engaging if it is to be used by agile teams; there is still an open question on when and how often to play or model (once that playing EoP in every sprint does not seem to be an option, as this would take quite a lot of time from each sprint); and, there is a need for an additional process to pick the key issues (risks, bugs, improvements) to address in the project after playing EoP. Additionally, a similar study was done on the risk estimation game Protection Poker (PP) [26]. PP includes activities on assets that are relevant for threat modeling, and it was found that identifying and prioritizing assets was considered useful by the teams, however challenges included the time it took, how to know if the granularity of assets is right, to know how to assess an asset's value and understanding the difference between asset value and exposure of a feature.

In this work we will base the analysis of our observations in the conceptual framework established by the previous literature [2], [25] and our previous work [4], [26] building evidence for some of the theoretical propositions of the previous work.

III. METHODOLOGY

A. Context

This study is part of a bigger project[1] which investigates how to meaningfully integrate software security into agile software development activities. The project started in October 2015 and will end in October 2020. The company of our case study is one which we have had long-term collaboration with since 2013. The method of choice for the project is Action Research [27], which is an appropriate research methodology for this investigation because of the combination of scientific and practical objectives that aligns with the basic tenet of action research, which is to merge theory and practice in a way such that real-world problems are solved by theoretically informed actions in collaboration between researchers and practitioners [27]. Canonical Action research is one of the many forms of action research [28], it is iterative, rigorous and collaborative, involving focus on both organizational development and the generation of knowledge. We apply canonical action research in this project [29].

Our study is based on an agile software organization spread over three different geographical locations. The company is a small/medium size organization with less than 100 employees in Norway, Poland, and Finland. There are mainly 5 product teams in this organization. They provide applications for fleet management, real time transport information systems and travel planning and ticketing systems. The solutions are complex in the sense that they involve hardware and software integrated solutions, as well as mobile and web applications.

Furthermore, action research involves intervention. As a result of our collaboration, the company has introduced the role of Security Champion[2] in each team, these are team members that are more focused on security aspects than the other team members. Security Champions facilitate and sometimes coach the security activities in the team. The Security Champion is not necessarily a security expert, but a developer/architect with an above average interest and aptitude for security. This role is similar to what McGraw refers to as "the Satellite" in the BSIMM[3]. The company has also a professional that is named Security Officer, and her main responsibilities are to provide leadership and guidance to the teams on the approach to adopt security activities in the teams and to perform oversight on the activities that are needed for assurance that the security is addressed properly in the company. The introduction of the threat modelling as part of the Software Development Life Cycle was also another intervention, as described below.

B. Our Approach to Threat Modeling in the Intervention

At the companies where we are doing threat modeling, we follow the strategy based on the Microsoft Threat Modeling framework [18]. We have found that it provides a structured, systematic approach to threat modeling, which may also explain why it is the most frequently used approach in Norwegian companies and elsewhere [22]. In our instantiation of the approach, a threat model is a visual representation of four main elements: Assets which are essential or critical for the system; a description of how assets are stored, processed or otherwise interact with the system (usually described as a Data Flow Diagram); the attack surface of the system; threats which will affect one or more of the identified assets.

Asset identification is the first step. An asset is something that needs to be protected within the system. Usually, assets are the information or services that are vital for the business operation and success, however, the concept of "assets" can also comprise other parts of the system, such as hardware, network components, domains or even people. Asset identification is often assumed to be done implicitly, but we have formulated an explicit method [3] which we have since evolved. Briefly, developers and the most important system stakeholders perform a semi-structured brainstorming session, to get all possible assets on the board. We then engage the participants in an interactive classification session where we aim to determine the assets' relative importance or value. Based on the developers' knowledge of the system, we finally ask the participants to determine the assets' relative ease of exploitation, and end up with a grid. After the meetings the team members were asked to annotate the asset list with more information about each asset.

The second step in the threat modeling exercise is to get an overview over where the assets are stored, processed or otherwise interact with the system. As part of this step, it is also useful to define the interfaces of the system under analysis and to identify potential attack surfaces. As a means

[1] http://www.sintef.no/sos-agile

[2] https://www.owasp.org/index.php/Security_Champions

[3] https://www.bsimm.com

to get an overview of the system, we recommend making a Data Flow Diagram (DFD). A DFD is a graphical representation of the most important actors, processes, services, components and data stored in the system; highlighting how information flows between each of them. The most frequently used tool for drawing DFDs is the Microsoft Threat Modeling Tool (TMT), which implicitly also makes it the most popular threat modeling tool; TMT encompasses all aspects of security to offer documentation as a guide through the remaining process. The attack surface is identified directly from the DFD by looking at the trust boundaries between the system and external entities.

The last step in the threat modeling exercise is to identify and analyze all relevant threats to the system. This can be done in several ways, but we performed it using the STRIDE framework [18].

STRIDE was used to identify threats by analyzing each of the interfaces defined in the DFD and assessing whether any of the attack types were relevant. Relevant attack types were studied further to determine how they could be executed or applied directly on the identified assets. *STRIDE is* based on the first letter of each of the attack types [18]:

- **S**poofing is forging a sender identity or attempting to access a system by using a false identity.
- **T**ampering is the unauthorized modification of data.
- **R**epudiation is the ability of users (legitimate or otherwise) to deny that they performed specific actions or transactions.
- **I**nformation disclosure is the unwanted exposure of private data.
- **D**enial of service is the process of making a system or application unavailable.
- **E**levation of privilege occurs when a user with limited privileges exploits system weaknesses to gain privileged access to an asset.

After the meetings we asked the Security Champions to create a list of risks based on the threats that were identified in the meeting.

C. *Data Collection and data analysis*

In the study presented in this paper, our aim was to investigate how to apply threat modeling in agile projects, and our focus was on the challenges the teams are facing. Our research questions are:

- What are the main challenges to applying threat modeling in agile software development?
- How can we adapt the approach to better suit agile software development projects?

To answer our research questions, we observed eight threat modeling sessions, each lasting typically 1-2 hours. The data collection was the observations after each session, and periodic discussion meetings. One of the authors conducted systematic meetings with the security champions every other week for 6 months, about different aspects of the job as Security Champion. It should be noted that for these teams this was the first time they were performing threat modeling, and some teams had to run more than one session to cover all parts of the DFD. Our observation template is reproduced in Fig.2.

We have also talked with the participants of the meetings during informal talks and bi-weekly discussions with the security champions of each team. Besides the observation files, we have also the following artefacts as results of the sessions:

- Updated data flow diagrams;
- Updated asset lists with prioritization;
- A list of threats based on STRIDE per covered asset;
- List of risks created from the sessions;

We coded the observations using the MaxQDA [4] qualitative data analysis tool, in an exploratory way so we could see which theme would emerge from the observations only. We first started generating the initial codes from the participants' quotes with the support of MaxQDA. After that we searched for themes among codes. Once themes were identified and listed, we grouped them by phases (preparation, execution and post-execution), having a total of 21 general main challenges encountered. We then classified these issues according to the challenges and, finally, the researchers interpreted and discussed the findings together in order to reach consensus.

We describe below the observations from each of the phases of the process: asset identification, creating the DFD, running the threat modeling meeting with STRIDE, and after the meeting. We then make an comparison with the challenges found by Gálvez and Gürses [25] to confirm or refute the challenges they have elicited.

IV. Results

Table 1 shows the 21 challenges to threat modeling that emerged from our observations. In **Table** 1 we do a mapping of the challenges from Gálvez and Gürses [25] and Tuma et al.[2] to what we have observed, showing if we have raised similar challenges or not, or if we have found contradictory results. We do not aim to compare, but try to find a mapping of the challenges (Table **2** and Table **3**).

Observation template

1 Observation

- Date and time:
- Technique:
- Observers and their roles:
- Group:
- Number of participants:
- How many times have they used the technique before?
- What questions did they have on the technique?
- Did they make or suggest any changes to the technique? (If so, explain)

2 Additional things to cover in the reflection afterwards

- To what extent did everybody participate?
- How was the general mood in the room?
- What worked well, what was challenging?
- What security topics were mainly discussed (threats, assets, vulnerabilities, mitigations)?
- How did the observation go?
- Any thoughts/opinions on how we as observers and participants may have influenced the process?
- Other reflections:

Fig. 1 - Observation Template

[4] https://www.maxqda.com/

Step	#	Challenge	Our Observations from Our case Study	Mapping to [25],[2]
Asset Identification	**C1**	**Documentation of the assets after the meeting was not done.**	We asked the teams to document the assets after the meetings, but most of the teams did not do this. When asking the teams afterwards, they mentioned that they did not see how this documentation would be useful. They thought that having the list of assets only with names and the prioritization and the discussion of the prioritization was good enough.	T1
	C2	**Many discussions on threats and mitigations strategies get lost**	It was very good to have the graph so people could discuss the exploitability of the assets, but at the same time at the meeting the team gets to discuss what are the mechanisms that they have to protect the assets, but once that the focus is on the assets, all this discussion gets lost.	T1
Data Flow Diagrams (DFDs)	**C3**	**It is challenging to motivate the teams to draw the diagrams**	Some security Champions did not feel motivated to write down the DFDs because they felt it was overwhelming and hard to have a good drawing of the design of the systems. Some teams spent a long time (more than 10 hours) creating the DFD and they mentioned it felt like a they were taking time from "development activities" that then impacted their productivity of "functionality output".	
	C4	**It was hard to decide the right level of abstraction to the DFDs.**	In some teams the DFD was too high level, which impacted the discussions, sometimes we had to draw a deeper view of the DFD during the meeting, leading to less effective meetings. We noticed that if the DFD is too complex, then there is a need for many meetings to cover the DFD. If the DFD is too simple, then the discussion gets unfocused and vague.	G2, T3
	C5	**It takes long time to draw the diagrams**	In most of the teams, the security champion was the one documenting the diagrams, but they needed input from different people in the team. Some teams spent many hours trying to draw a good DFD.	T1
	C6	**It was challenging to map the interfaces with other teams**	Some teams had problems with drawing the interfaces with other internal systems in the company because they are a company focused on product development teams, but have projects that comprise many products. They were also not sure how to deal with cross-cutting concerns. It was not easy to get all teams in the same meeting, or to know who to invite from the different teams to the discussion.	G3, T3
	C7	**The approach does not make a link with the actual code**	The teams mentioned that they actually did not know how the system was really implemented, and the DFDs did not give that confidence that the system was actually implemented that way, especially where there was a lot of legacy code.	G4, T2
	C8	**It is challenging to maintain the DFDs.**	Because of the lack of focus on documentation in agile teams, the teams did not have a list of assets or a DFD of the system ready, and we had to ask them to create these artefacts for the meeting, and they did not show motivation to maintain or have set a strategy to update the DFD. The teams did not feel like this was an activity that they would want to do frequently. The main impression was that they would do it now, and then maybe in one year or something. It was not easy for the teams to foresee how often they would need to perform a new threat modeling for their systems. We as researchers also had problems to state clearly how to decide.	G1
Modeling Meeting	**C9**	**The meeting needs to be structured but it is not always clear on how to run the meeting.**	We tried different approaches to perform the meeting: based on covering the DFD; based on covering STRIDE; or ad-hoc based on where people thought the focus of the meeting should be. Sometimes we changed the approach as we saw it was not working well with a specific team. We still do not have clear recommendations on how to choose an strategy for the meeting.	T1
	C10	**It is hard to decide which other people should be included in the meetings besides the "core" development team.**	Specifically when trying to do the threat modeling across products, it was not easy to identify who should be the representative from each product. We also had some questions on whether or not we should include the Product Owners in the meetings. One product owner was very helpful in the meeting, but at the same time made the discussion very focused on his view of the system and the other team members did not participate as much. We noticed that for STRIDE types of discussions it was very beneficial to have the operations representative in the meeting.	G3
	C11	**There are challenges on cross-products modeling**	We created a meeting that we named "Cross Products Threat Modeling" to try to identify problems that one product could create for another product. We had many challenges with that meeting: to structure, to define which model to use for the analysis, to motivate people to be in the meeting and to decide who should be in the meeting. This is one issue that we have not found a satisfactory solution for.	G16, T3

	C12	**There are challenges with running meetings in distributed settings.**	Distributed teams are used to using video conferencing to replace or complement physical meetings, but we have found that this is challenging when doing threat modeling: there are still some technical challenges with the videoconferencing equipment; it is generally not possible to see the people at all location(s) and their whiteboard(s) at the same time, and often not even the whole whiteboard at a resolution that allows everyone to read everything. We have experimented with different configurations, but the major challenges remain unresolved. We have tried making identical drawings at each participating site, but consistency is hard. When doing asset analysis we have tried writing notes and diagrams locally, letting remote participants provide input verbally, and we tried doing it the other way around, but there seems to be drawbacks with both solutions.	
	C13	**It is hard to know when enough analysis has been done.**	Because there always is the sense of "ongoing" project with agile, it is hard to explicitly say, "now it is good enough for the time being". It is also hard to say for how long this analysis is valid. One Product Owner mentioned that this is "useful rehearsal which should be done regularly in all projects". Another participant also mentioned that "It helps to document the mitigation strategies that are in people's head, but at the same time, it is hard to know if we have extracted all knowlegde needed." As mentioned before, covering the whole DFD will often take too long; in some cases people get bored of the meetings; they feel like they have to follow up first the risks and threats elicited before discussing more about threats in a new meeting; at the same time it is not so straightforward to create a list of actionable items from the meetings.	G5, T4
	C14	**The meetings are not effective**	To agile teams, 1-2 hour meetings means a long time taken from the sprint hours. Furthermore, when they saw that they did not cover the full DFDs, the team members got somehow frustrated with the process. The coverage of the DFDs per meeting was much lower than we expected. We could not run the meetings with the teams indefinitely, so we decided to stop after two or three sessions with each team.	G13, T1
	C15	**There is a need for a Security Expert to run the meeting; not every team has this professional available**	Most agile teams nowadays do not have a "security expert" and this is a challenge. In our case study, the second author was the one running the meetings, and the other researchers doing observations. The security expert helps to facilitate and to leverage the discussion. The security champion was most of the times documenting the threats.	Contradict G10
	C16	**It is not easy to have everyone participating**	In some meetings, if there is someone that is more "expert", "more experienced" or "more knowledgeable" in the project, he may take over the discussion, and the participation of the others diminishes. There is also the problem that not every one is expert in all parts of the DFDs, then in some meetings some people are not able to participate.	
STRIDE	**C17**	**STRIDE focuses too much on the "communication channels"**	We noticed that many times that STRIDE was limiting the discussion to the trust boundaries and the communication channels, thereby neglecting potential threats relevant to other parts of the product.	G12
Outputs from the Sessions	**C18**	**The output of the sessions are a list of concerns/ threats that are not concrete**	As a follow up, we asked the Security Champions from each team to formalize the threats discussed in the meetings as risks. The Security Champions did not feel completely confortable to write them down, and sometimes we needed to ask them many times about it. The meeting also did not focus much on the decision of impact and probability of the threats or on which mitigation actions would be done, because it would take too long. This was a follow-up that the Security Officer had to do with the security champions. We also asked the team to contribute to the documentation of the threats, but most teams did not prioritize this.	
	C19	**Followup of the threats is challenging**	As mentioned in C18, we had a challenge to make the security champions follow up and describe the threats more in detail after the meeting was done.	
	C20	**The list of threats creates a concern on time.**	The team members were not sure when they would have time to prioritize threats from the parts of the product that they thought had passed the "definition of done".	
	C21	**Not finding threats gives a false sense of security**	When the team had a very good design upfront and all the security issues were thought through in a good design it gives then a sense that the product is secure enough, and that they thus do not need to worry about security.	

Table 1 - Challenges identified in this Case Study

V. Discussion

A. Implications to Research

The study reported in this paper provides evidence on challenges that agile teams face when conducting threat modeling meetings in a software company with five different teams. Coincidentally, both our study and the study by Gálvez and Gürses identified 21 challenges each, with some overlap (as seen in Table **2**). We could observe 9 of their 21 challenges (G1, G2, G3, G4, G5, G12, G13, G15, G16). The extra challenges we have observed in our exploratory study came mostly from the activities of asset identification and also from our observations on what happened after the threat modeling meetings.

It was surprising to us that Galvez and Gürses observed that bringing experts into the development process slows it down (G10), because in our case the security expert helped to facilitate the process. It is important to note that the agile teams we followed did not have a security expert prior to our intervention. Also, the expert helped to drive the dicussions where there was bigger probability of finding threats to the systems. Further research is needed to study the effect of the security expert on the threat modelling process.

For different reasons, we were not able to observe nine of the challenges (G6, G7, G8, G11, G17, G18, G19, G20, G21):

- G6 (Threats can emerge, change or vanish as the system evolves) and G17 (Strategic mitigations are difficult to spot in an evolving system): we did not observe these phenomena in the teams because we did not observe what happens over a longer period;
- G7 (Requirements elicitation from customers is slow): in our context the threat modeling was performed during maintenance phase of the products;
- G8 (Customers may not possess enough privacy knowledge), G11 (Business goals are difficult to translate to privacy requirements) and G18 (Conflicting privacy requirements for the same service): we did not focus on privacy, and, in our meetings the customers were not involved;
- G19 (No common infrastructure to enforce rules across different services), G20 (Lack of information to automate testing) and G21(Manual validation requires expertise and documentation) were not the focus of our analysis.

Our study also confirms all the challenges found by Tuma et al.. Some challenges we found overlap with some challenges from the literature and some were new. Many of the new challenges elicited in this study (C18-C21) were also found in the study we did previously of EoP and of Protection Poker in a capstone project, thus they are likely not specific for the company we studied (see Table 5 for an overview of findings from that study that are relevant to the challenges we have identified in this study).

When mapping the challenges to the agile manifesto (Table **4**), we can see that our study showed challenges mostly influenced by the focus on "individuals and interactions over processes and tools" and on "working software over comprehensive documentation". One of the reasons for this in our study is that we focused on the meetings, and the challenges we had on running the meetings.

Table 2 - Mapping of Challenges from Gálvezand Gürses [25].

Challenges from Gálvezand Gürses [25]	**Our Study**
G1: Up to date model	√
G2: Vague requirements descriptions	√
G3: Modular diagram	√
G4: Reflect implementation details into the system diagram	√
G5: Keep the threat list up to date	√
G6: Threats can emerge, change or vanish as the system evolves	
G7: Requirements elicitation from customers is slow	
G8: Customers may not possess enough privacy knowledge	
G9: The attacker perspective may not lead to realistic threats	
G10: Bringing experts into the development process slows it down	X
G11: Business goals are difficult to translate to privacy requirements	
G12: Threat catalogs are limited	√
G13: Deriving threats is slow	√
G14: Analyzing scenarios requires a lot of creativity	
G15: Performing risk assessment is slow	√
G16: Finding cascade failures from combinations of services is difficult	√
G17: Strategic mitigations are difficult to spot in an evolving system	
G18: Conflicting privacy requirements for the same Service	
G19: No common infrastructure to enforce rules across different services	
G20: Lack of information to automate testing	
G21: Manual validation requires expertise and documentation	

Table 3 - Mapping of Challenges from Tuma et al. [2].

Challenges from Tuma et al. [2]	**Our Study**
T1: Automated Threat Analysis	√
T2: Propagation to the Source Code	√
T3: Need Guidelines for Composition Analysis.	√
T4: Need Guidelines for Definition of Done	√

Table 4 – Mapping the Agile Manifesto Principles to the Challenges.

Agile Manifesto	**Tuma et al.** [2]	**Gálvezand Gürses** [25]	**Our Study**
Individuals and interactions over processes and tools		G3	C9, C10, C12, C14, C16
Working software over comprehensive documentation	T2	G2,G3, G4, G20, G21	C1, C3, C4, C5, C6, C7,C8, C11, C19, C20
Customer collaboration over contract negotiation		G7, G8, G9, G10, G11	
Responding to change over following a plan	T1, T2, T4	G1, G6, G5, G15, G16, G17, G18, G19	C8, C13

Table 5 – Challenges also observed in our previous studies in a study of Protection Poker [4] and Microsoft EoP [26] with University Capstone Projects.

Selected challenges from Tøndel et al. [4], [26]	**Our Study**
Asset value was mixed up with exposure of feature in the discussions	C2
Students did not use the details of the DFD in the discussions.	C4
Students played EoP early in the project, and it was unclear how the design and code would be.	C7
Students ended up having varying preferences on whether to do threat modeling as a game or as a checklist.	C9
Students did not play the full EoP card deck, but students considered themselves finished anyway.	C13, C14
The security expert was essential to understand the threats (hints on the cards) and how they apply to the system.	C15
No vibrant discussions due to limited security knowledge. Some were more knowledgeable and more active than others.	C16
Many of the existing cards were not considered relevant.	C17
Score sheets ended up having few details. It was not clear to the students what to do with the results after the EoP session.The students wanted more support on mitigations actions. No impact on the design and implementation choices were observed.	C18
Results from the sessions were not followed up	C19
Some groups were instructed by customers to drop the security considerations in order to prioritise functionality.	C20
Playing parts of the deck and not finding much of relevance made them reluctant to look at the remaining threats.	C21

Individuals and interactions over processes and tools was challenging specially because it was hard to get effective meetings with clear and actionable outputs and because in some teams they were in a distributed setting which made the meetings even harder to run. We also see a need for better guidelines on how to run the meetings. In our previous paper on running EoP (Elevation of Privilege) with teams of students, we have found that there are many problems with running the threat modeling meetings using EoP [4]. Therefore more studies needs to be done in order to understand what is the best process to run these meetings in an agile context. We can see with the challenges that there are needs for guidelines on what to document, which level of details the diagrams need to have, how to structure the meeting to be motivating and effective and how to document and follow up the output of the meetings.

The focus on working software over comprehensive documentation is a challenge for security work, as, it is many times very much based on the documentation of the decisions, risks and assets. We have yet not found very good arguments to motivate the teams and show them the advantages of having all this extra documentation done.

Gálvez and Gürses [25] have found six challenges on the principle of customer collaboration, but we have not observed these. In our case we did not experience these challenges because the context of when we had the threat modelling was during maintenance phase of the products, and the focus of the threat modelling activity was on the product and not on the specific projects. We have tried to have the Product Owners in some meetings, and this was mostly beneficial rather than challenging, especially because it created more awareness of the threats, and also because they could understand more of the impacts and exploitability of the threats. As future work we will also have meetings in contexts of the projects and involve the customers in the discussions.

On the principle of responding to change over following a plan, we have found two challenges related to the maintenance of the DFDs, and the question of deciding when enough analysis has been done. Tuma et al. [2] have also identified this problem and propose that a "definition of done" needs to be further explored for activities in threat analysis.

Further research should address the benefits of having security documentation in agile projects. One way to address this is to focus on the insights provided by Tuma et al. . [2] that existing analysis techniques have yet to mature in traceability of analysis in the code base, composability of analysis outcomes and threat impact analysis automation;.

B. Implications to Practice of Threat Modelling with Microsoft STRIDE and DFDs.

On the implications to the practice, we observed in this study that there are many improvement points on the approach for threat modelling using DFDs and STRIDE. Companies must be aware of these and apply the technique accordingly.

One recommendation to the practitioners is to include a security expert for facilitating the meeting, helping the meeting to be more focused, and also more relevant to the participants. As we mentioned before, this company has adopted the role of Security Champions in each team, and they were very important players in the whole process; they were the ones driving the drawing of the DFDs, scheduling the meetings, documenting the threats found and creating and following up the risks identified in the meetings. Clearly, they need to be "coached/trained" to build the skills needed to perform these activities, and our role as researchers doing action research with them, influenced this process and helped them to get the needed skills.

The observations shown in this study also lead us to affirm that that any extra activity that is performed to address security concerns, will potentially slow down the process of creating functionality, and if the developers are only considering the features output, without being concerned about the quality of these features we may be able to say that performing threat modeling may slow down the development process, still we would say that the company in this study see the benefits of keeping the practice and will keep adding the activity as part of the regular and systematic approach to security.

There is also a need to be better on making clear to the teams about the benefits of doing threat modeling for the project, while acknowledging the possible impacts in time and costs, not only for the meetings themselves, but also on the time needed for preparation of the meetings and follow-up of the outputs of the meetings. In addition, the side effects of the activity should be highlighted, such as: better documentation of the system, awareness of security issues for all team members, better confidence on the way security is addressed in the team, and better visibility of the threats to other stakeholders such as Product Owners and possibly managers

and executives.

We also noticed a decrease of motivation from the asset identification to the threat modeling meeting. One hypothesis we have is that the first meeting on the assets created more awareness, and they were excited to learn something new; the threat anslysis meeting focused more on trying to document the threats, and it was not as engaging.

VI. Limitations

In any action research study, the role of the participating researchers may have a significant influence on the result. In our case, we believe that since there was little software security expertise present in the organization under study, it is likely that it might not even have attempted to conduct any threat modeling exercises without us. We will maintain our collaboration with this organization in the years to come, in order to learn whether the threat modeling activities that we have initiated can become self-sustaining.

Our participation in the process may have also influenced the results in the sense that it is possible that either we have not found all challenges that other companies are facing, or we have introduced challenges by our participation in the process. The participant observer role of the researchers has been more active than the usual case, since we facilitated the meetings and also assisted the Security Champion in the role of a security expert. However, we believe that in the long run, the security champions at the studied company will acquire enough security modeling expertise for performing this activity with the team. Not every developer can be a security expert, but every software development organization needs to have enough expertise to conduct most of the software security activities in the development lifecycle.

Common criticisms to a case study also apply to this study, among them one may list: uniqueness, difficulty to generalize the results, and the introduction of bias by participants and researchers [30]. In our study, we generalized the findings from empirical findings to theoretical statements, which involved generalizing data from collected data and perceptions by discussing them in accordance with the literature. Observation data were our primary source of information, and therefore they have the limitations of the possible research bias.

Qualitative findings are highly context- and case-dependent, and this is also true for our study. We sought to mitigate this by analyzing activities in five product teams. All the participants were professionals using typical development technologies in a typical working environment, e.g., the natural setting demanded by the case study approach. We described the main characteristics of the case study, including context and settings, data collection, and analysis process. We believe that this makes the results easier to generalize.

As commonly done in in-depth qualitative studies, we also had to do a trade-off between the number of participants, the duration and the cost of this study. The number of projects and variance in the context is not quantitatively significant, but gives deeper insights on the issues investigated in this work.

VII. Conclusions and Future work

Introducing software security activities in an agile development lifecycle does not come for free; by necessity, extra activities require extra time and effort. The research results presented in this paper contribute to the body of knowledge in applying security activities in the agile context. Challenges were described related to the different principles of agile.

Some challenges overlap the findings from the literature and the extra challenges we have found in our exploratory study came mostly from the activities of asset identification and also from our observations on what happened after the threat modeling meetings.

We have also identified some lessons learned for companies that wish to perform threat modelling using the Microsoft Threat Modelling approach.

This study shows that we still have to address many challenges in order to get a proper adoption of threat modeling in agile development projects. Therefore, as future work, our goal is to better understand the challenges here elicited, and validate our findings in different contexts and organizations we are collaborating with in the current project.

Acknowledgment

This work was supported by the SoS-Agile project: Science of Security in Agile Software Development, funded by the Research Council of Norway (grant number 247678).

References

[1] G. Mcgraw, “Software security,” *IEEE Secur. Priv. Mag.*, vol. 2, no. 2, pp. 80–83, Mar. 2004.

[2] K. Tuma, G. Calikli, and R. Scandariato, “Threat analysis of software systems: A systematic literature review,” *J. Syst. Softw.*, vol. 144, pp. 275–294, 2018.

[3] M. G. Jaatun and I. A. Tøndel, “Covering Your Assets in Software Engineering,” in *2008 Third International Conference on Availability, Reliability and Security (ARES)*, 2008, pp. 1172–1179.

[4] I. A. Tøndel, T. D. Oyetoyan, M. G. Jaatun, and D. S. Cruzes, “Understanding challenges to adoption of the Microsoft Elevation of Privilege game,” in *HotSoS*, 2018, p. 2:1-2:10.

[5] C. R. Camacho, S. Marczak, and D. S. Cruzes, “Agile team members perceptions on non-functional testing influencing factors from an empirical Study,” in *Proceedings of the 11th International Conference on Availability, Reliability and Security, ARES 2016*, 2016, pp. 582–589.

[6] T. D. Oyetoyan, D. S. Cruzes, and M. G. Jaatun, “An empirical study on the relationship between software security skills, usage and training needs in agile settings,” in *Proceedings of the 11th International Conference on Availability, Reliability and Security, ARES 2016*, 2016, pp. 548–555.

[7] M. G. Jaatun, D. S. Cruzes, K. Bernsmed, I. A. Tøndel, and L. Røstad, “Software Security Maturity in Public Organisations,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2015, pp. 120–138.

[8] G. McGraw, B. Chess, and S. Miques, “Building security In maturity model (BSIMM 8),” *http://bsimm.com.* 2017.

[9] T. D. Oyetoyan, M. G. Jaatun, and D. S. Cruzes, “A Lightweight Measurement of Software Security Skills, Usage and Training Needs in Agile Teams,” *Int. J. Secur. Softw. Eng. IJSSE*, vol. 8, no. 1, pp. 1–27, 2017.

[10] M. Almorsy, J. Grundy, and A. S. Ibrahim, “Automated software architecture security risk analysis using formalized signatures,” in *35th International Conference on Software Engineering (ICSE)*, 2013, pp. 662–671.

[11] R. Scandariato, K. Wuyts, and W. Joosen, “A descriptive study of Microsoft’s threat modeling technique,” *Requir. Eng.*, vol. 20, no.

2, pp. 163–180, 2015.

[12] M. Cagnazzo, M. Hertlein, T. Holz, and N. Pohlmann, “Threat modeling for mobile health systems,” in *IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, 2018, pp. 314–319.

[13] M. Abomhara, M. Gerdes, and G. M. . Køien, “A STRIDE-Based Threat Model for Telehealth Systems,” in *Norsk informasjonssikkerhetskonferanse (NISK2015)*, 2015, no. November.

[14] R. Khan, K. McLaughlin, D. Laverty, and S. Sezer, “STRIDE-based threat modeling for cyber-physical systems,” in *IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, 2017, pp. 1–6.

[15] M. G. Jaatun, I. A. Tondel, and M. Bartnes, “Threat Modeling of AMI,” in *CRITIS 2012*, 2013, no. January, pp. 264–275.

[16] C. Mockel and A. E. Abdallah, “Threat modeling approaches and tools for securing architectural designs of an e-banking application,” in *Sixth International Conference on Information Assurance and Security*, 2010, pp. 149–154.

[17] M. M. Aydin, “Engineering Threat Modeling Tools for Cloud Computing,” University of York, 2016.

[18] A. Shostack, *Threat modeling: Designing for security*. John Wiley & Sons, 2014.

[19] P. Torr, “Demystifying the Threat-Modeling Process,” *IEEE Secur. Priv. Mag.*, vol. 3, no. 5, pp. 66–70, Sep. 2005.

[20] M. S. Lund, B. Solhaug, and K. Stølen, *Model-Driven Risk Analysis*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[21] T. Ucedavélez and M. M. Morana, *Risk Centric Threat Modeling*. Hoboken, NJ, USA: John Wiley & Sons, Inc, 2015.

[22] N. R. Mead, F. Shull, K. Vemuru, and O. Villadsen, “A Hybrid Threat Modeling Method,” Carnegie MellonUniversity - Software Engineering Institute - Technical Report - CMU/SEI-2018-TN-002, 2018.

[23] S. Türpe and A. Poller, “Managing security work in scrum: Tensions and Challenges,” in *SecSE@ESORICS 2017*, 2017, pp. 34–49.

[24] A. Poller, L. Kocksch, S. Türpe, F. A. Epp, and K. Kinder-Kurlanda, “Can Security Become a Routine?,” in *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing - CSCW ’17*, 2017, pp. 2489–2503.

[25] R. Gálvezand S. Gürses, “The Odyssey: Modeling Privacy Threats in a Brave New World,” in *IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, 2018, pp. 87–94.

[26] I. A. Tøndel, M. G. Jaatun, D. S. Cruzes, and T. D. Oyetoyan, “Understanding challenges to adoption of the Protection Poker software security game,” in *SECPRE 2018*, 2018.

[27] D. J. Greenwood and M. Levin, *Introduction to Action Research: Social Research for Social Change*. SAGE Publishing, 2007.

[28] R. Davison, M. G. Martinsons, and N. Kock, “Principles of canonical action research,” *Inf. Syst. J.*, vol. 14, no. 1, pp. 65–86, Jan. 2004.

[29] D. S. Cruzes, M. G. Jaatun, and T. D. Oyetoyan, “Challenges and approaches of performing canonical action research in software security,” in *Proceedings of the 5th Annual Symposium and Bootcamp on Hot Topics in the Science of Security - HoTSoS ’18*, 2018, pp. 1–11.

[30] D. S. Cruzes and L. ben Othmane, “Threats to Validity in Empirical Software Security Research,” in *In: ben Othmane, L. (Ed.), Jaatun, M. (Ed.), Weippl, E. (Ed.). (2017). Empirical Research for Software Security. Boca Raton: CRC Press.*, .